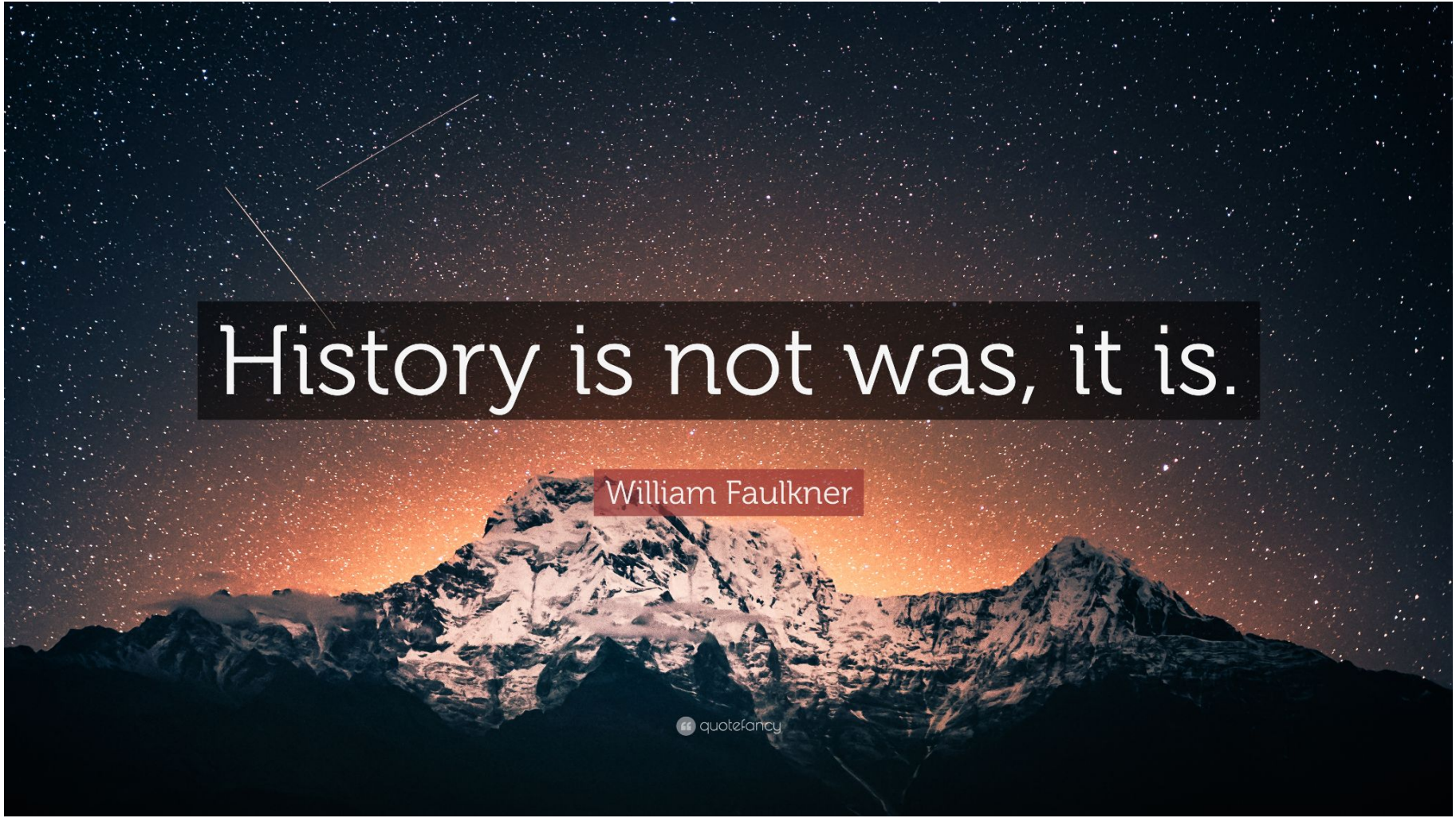
Unix Command Line

Kameswari Chebrolu
Department of CSE, IIT Bombay

https://pbs.twimg.com/media/E-YJGozUUAA6rUU.jpg

Outline

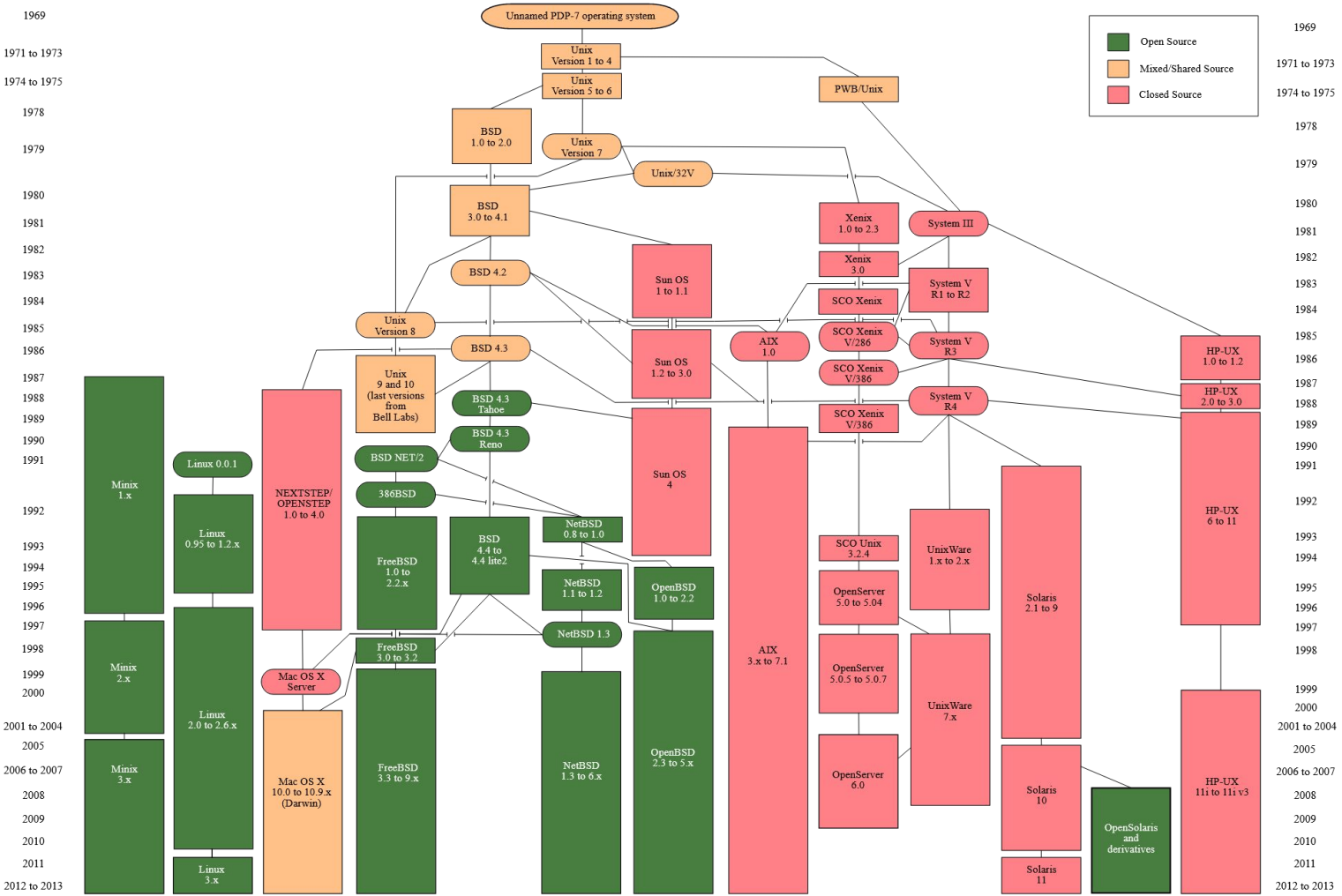
- Unix history and why popular?
- Command line vs GUI
- What is a Shell?
- Linux File System
- Various commands


History is not was, it is.
William Faulkner
quotefancy

Unix/Linux OS

- Unix: Proprietary OS created in late 1960s at AT&T Bell Labs
- Linux: a clone of Unix, free and open source
  - Written from scratch by Linus Torvalds in 1991

- Distributions of Linux: Linux OS packaged with lot of additional free software
  - Fedora, Ubuntu, CentOS, SuSe etc
  - Differ wrt to desktop environment, package installation, display server etc
  - Other Unix clones: FreeBSD and Mac OS X (its kernel Darwin, is based on BSD)
- A user on one Unix system can move to another easily wrt to command-line



https://sosheskaz.github.io/technology/2017/05/12/Adventures-In-Bsd.html

Popularity of *nix

- “Since we are programmers, we naturally designed the system to make it easy to write, test, and run programs” – Unix Creators, Dennis M. Ritchie and Ken Thompson
  - Very server and programmer-friendly OS
  - Linux (FREE) is for developers!
  - Easy to do scripting
  - Lot of scientific libraries and programs are written for *nix

- Open source (some versions) and exposes you to an ecosystem of open-source software
  - Helps bridge the concepts you learn with how they're applied in practice.
    - Interested in OS? Dig into details of open source linux and interaction with device drivers
    - Interested in Compilers? Clone gcc source
    - Interested in distributed systems? Clone Hadoop and run a cluster on your laptop
    - Interested in cloud computing? Containers origins in linux

Command Line vs GUI

Windows GUI: use pre-programmed interface ⇒ set of possible actions pre-decided

```
chebrolu@silmaril: ~/web-development-demo
chebrolu@silmaril:~$ mkdir web-development-demo
chebrolu@silmaril:~$ cd web-development-demo/
chebrolu@silmaril:~/web-development-demo$ mkdir dir1 dir2 dir3
chebrolu@silmaril:~/web-development-demo$ ls
dir1  dir2  dir3
chebrolu@silmaril:~/web-development-demo$ mkdir
mkdir: missing operand
Try 'mkdir --help' for more information.
chebrolu@silmaril:~/web-development-demo$
```

Command-line Shell: a prog. (scripting) language ⇒ use pre-written programs **AND** compose new scripts!

Power of the Shell

Alias: shell, terminal, console, prompt etc

1. Rename a set of files
2. Number of lines in all C files in a directory
3. Top five files with maximum number of lines

Demo!

A Brief History of the Shell

- Unix: OS for mainframe computers
  - Users connecting remotely via individual terminals (keyboard and screen)
  - No local programs, send text and receive text
  - Terminals based on text since text is light on resources
  - Commands kept very terse to reduce the number of keystrokes needed

- Need to support all kinds of file management tasks
  - Create files, list files, rename, move to folders etc
  - Each task required its own program (or command)
  - **Master program to coordinate execution of all these programs → shell**
- Original Unix shell called sh (Bourne shell)
  - Extended with better features and syntax is BASH (Bourne Again SHell)
  - Other shells also: zsh (mac OS), csh, fish etc

Basic Instructions

- Open shell: Click on “Activities” top left of the screen + type shell in the search box (or) use Ctrl-Alt-T
- Type a command in the same line as where $ (prompt) appears (command line ;-)
- Commands sometimes have number of arguments (command-line arguments)
  - tar -zcvf lab1.tgz lab1/

- The shell does not execute commands until the “Enter key” is pressed
- Any output the shell produces will usually be printed directly in the terminal
  - Another prompt is shown once finished
- Commands are case sensitive (ls vs LS)

Demo!

my folder : `Downloads`

me : `cd downloads`

Linux :

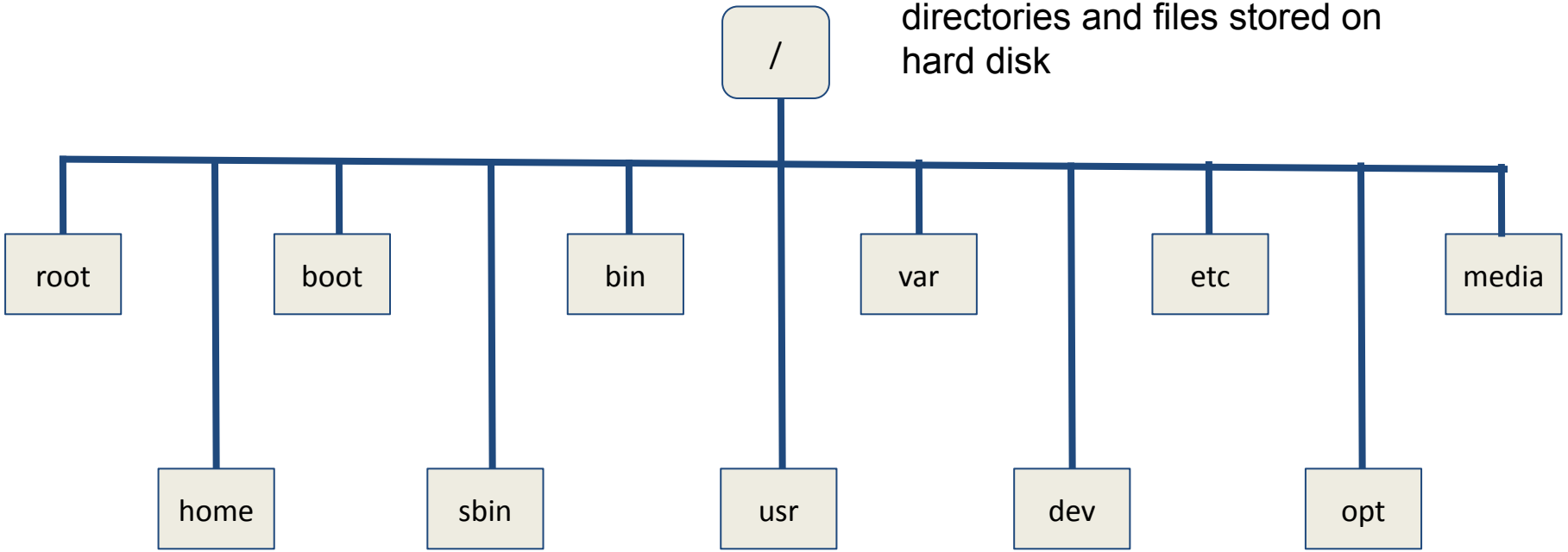
https://preview.redd.it/3zoe90gmq7y51.jpg?auto=webp&s=791da8fa068769a9e73f494f63e730c7f354fb86

Outline

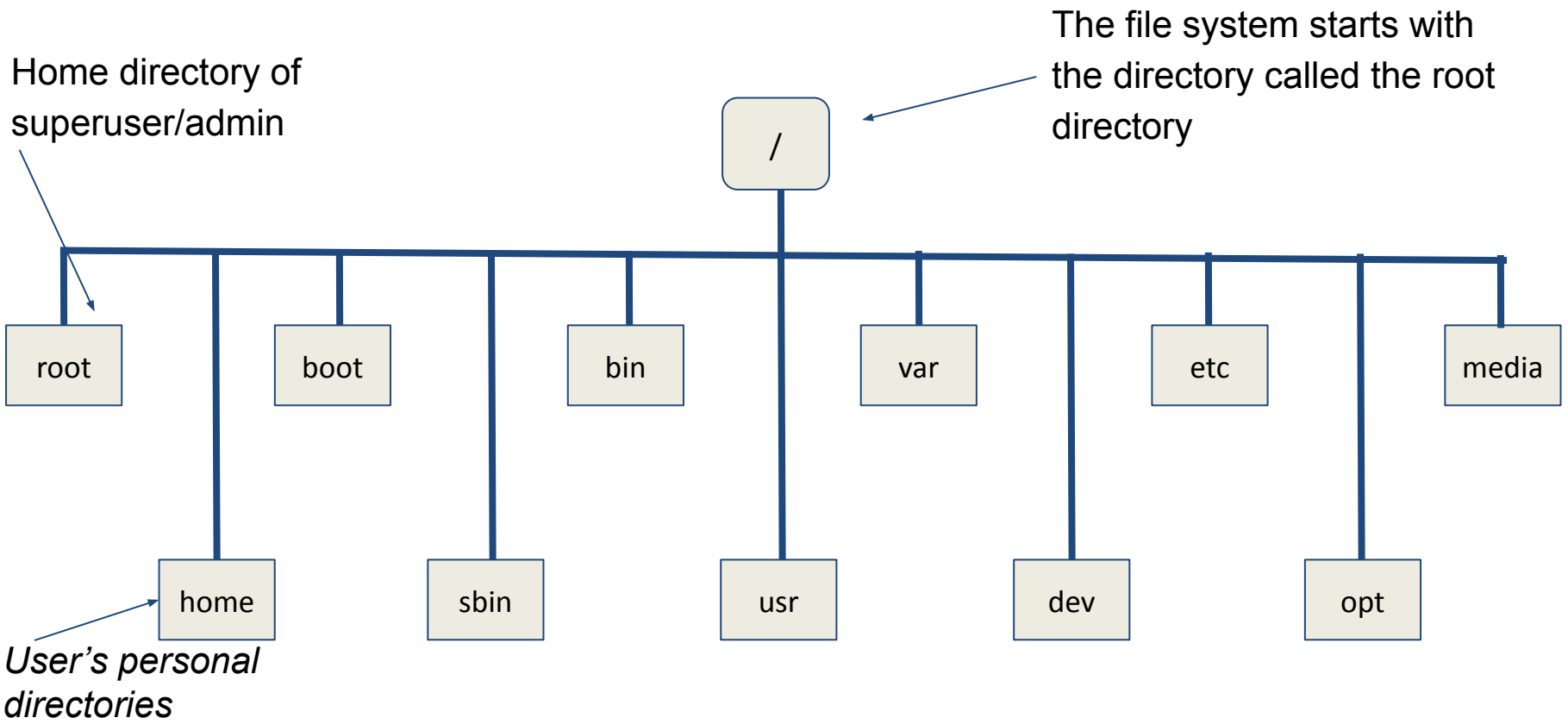
- ~~Unix history and why popular?~~
- ~~Command line vs GUI~~
- ~~What is a Shell?~~
- Linux File System
- Various commands

Linux Filesystem

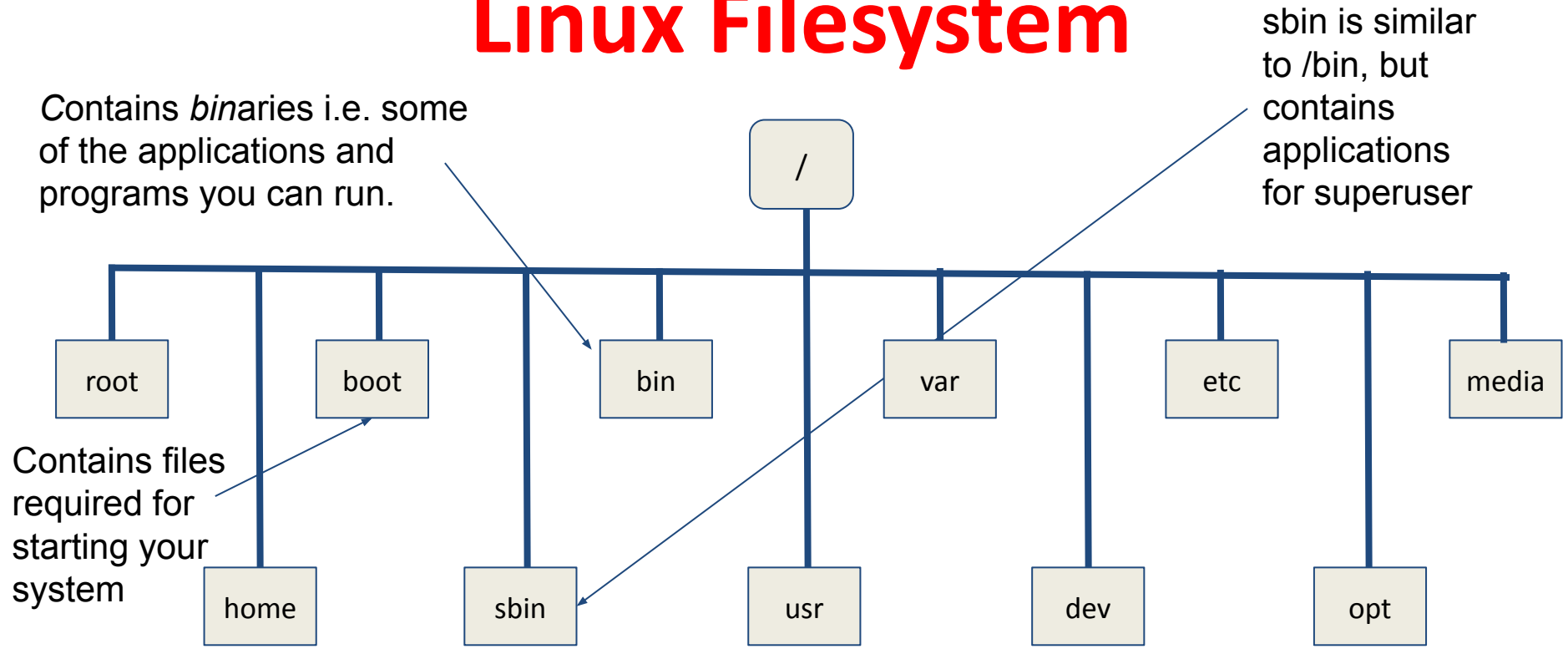
A filesystem is a hierarchy of directories and files stored on hard disk



Linux Filesystem



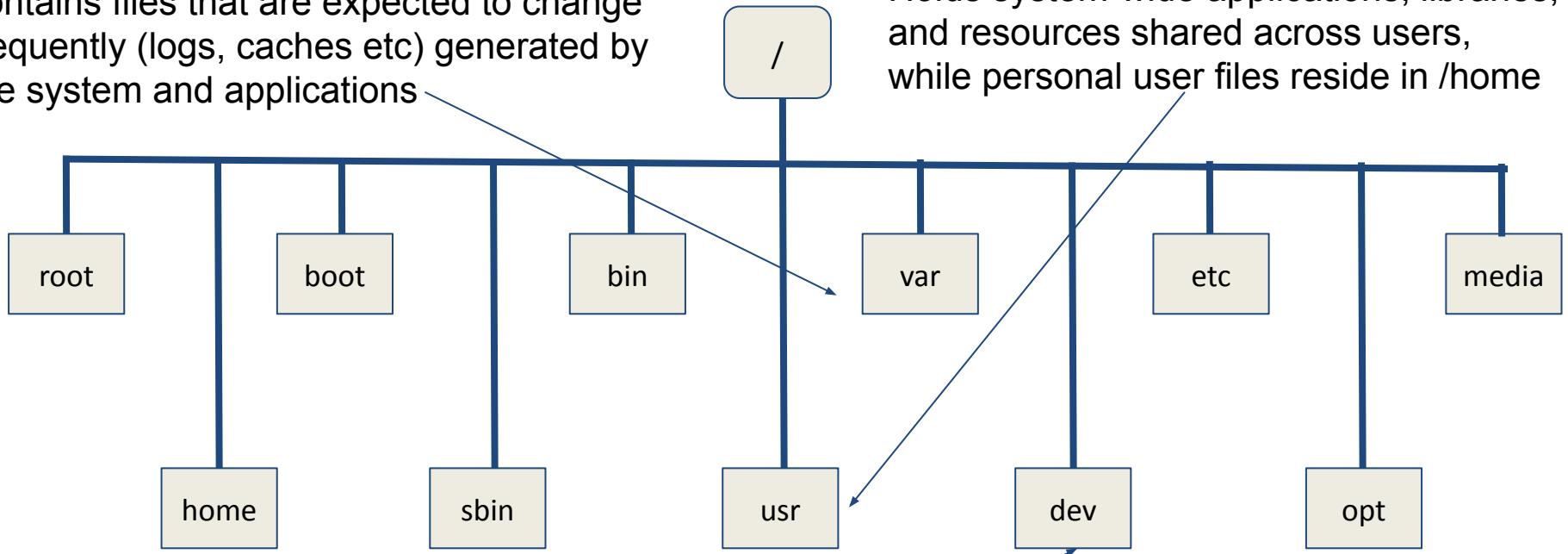
Linux Filesystem



Linux Filesystem

contains files that are expected to change frequently (logs, caches etc) generated by the system and applications

Holds system-wide applications, libraries, and resources shared across users, while personal user files reside in /home

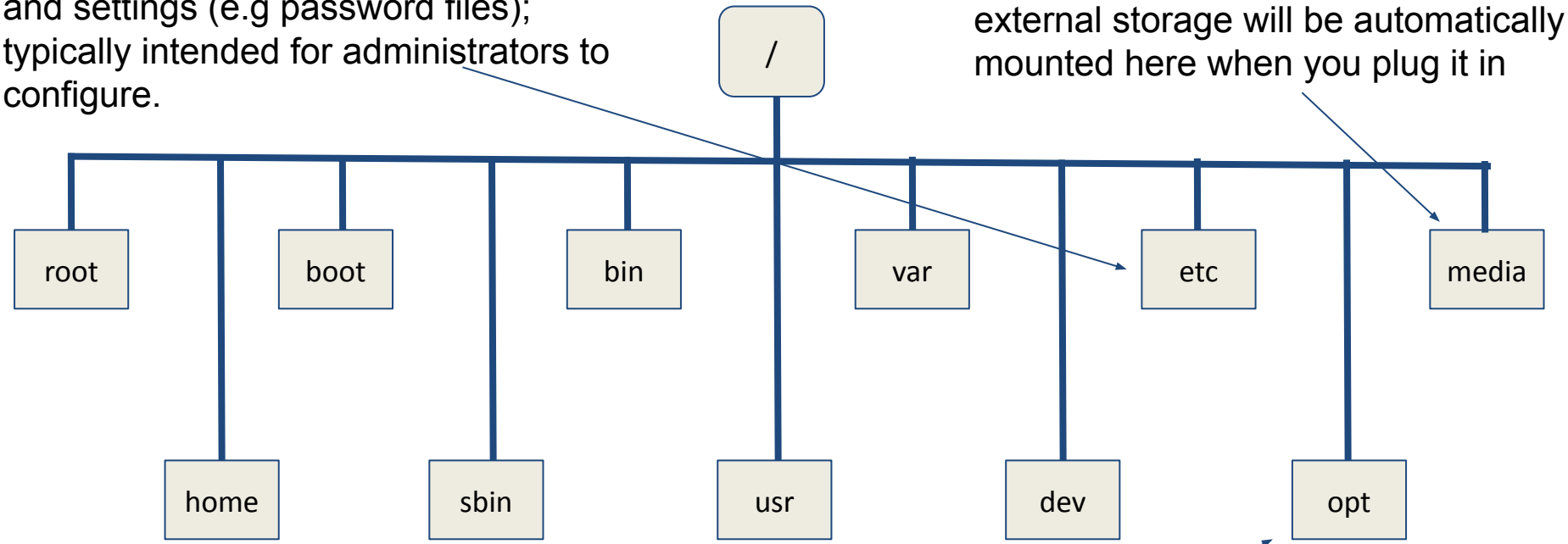


Contains device files (hard drives, USB devices, webcams etc). These files allow software to interact with hardware as if they were standard files

Linux Filesystem

holds system-wide configuration files and settings (e.g password files); typically intended for administrators to configure.

external storage will be automatically mounted here when you plug it in



optional or third-party software packages that aren't part of the default Linux distribution

Outline

- File and Directory Commands
- File Viewing and Editing Commands
- Commands for File Analysis
- Process Management
- Security and Permissions

File and Directory Commands

- clear
- man
- pwd
- ls
- cd
- mkdir
- rmdir
- cp
- mv
- rm

- These commands enable users to
  - Navigate the file system
  - Create, move or remove files and directories
- Provide a powerful interface for interacting with the operating system

Clear

- Clears the terminal screen
  - Terminal cursor moves to the top-left corner
- Helps enhance readability
  - Use before running new commands to avoid clutter and improve focus
- Note: Doesn't delete history or affect running programs
  - Only affects visual display

man

- Displays manual (help) pages for Unix commands
- Useful when learning new commands or options you are unfamiliar with
- Provides detailed documentation
  - Descriptions, options, usage examples, and technical details

- Usually formatted with a consistent structure
  - NAME, SYNOPSIS, DESCRIPTION, OPTIONS, EXAMPLES, SEE ALSO.
- Use the arrow keys or page up/down to scroll through the manual
- Press / followed by a keyword to search within the manual page
- Press q to quit the manual page

- Syntax: man [command]

pwd

- Shell has a notion of a default location
  - For the root user, home is at /root
  - Regular users, it is /home/username (e.g. /home/chebrolu)
- pwd (present working directory) command tells your current working directory
  - No options needed
  - Displays the full path of the directory you are currently in

- Use Case:
  - Helpful when navigating directories
    - Use pwd to confirm your current directory, especially when working in deep or complex directory structures
  - Helpful with scripting and automation
    - Dynamically get the current directory and perform operations relative to it

Demo

man, clear, pwd

ls

- ls: display contents of the current directory
  - Directories often listed in a different color (e.g., blue)
  - Executable Files may be displayed in green
- Syntax: ls [Options] [Files/Directories]
- Use Case:
  - Quickly see what files and directories exist in your current or specified directory
  - Checking file details like permissions or file size

- Key Options:
  - -l : Shows detailed information
    - File permissions, number of links, owner, group, size, and modification date
  - -lh : Displays file sizes in a human-readable format (e.g., KB, MB)
  - -lt : Sorts the output by the time of last modification, with the newest files first
  - -a : display all files including the hidden files
    - Every directory has at least two entries: “.” and “..” (called dot and dotdot)
      - dot directory is a shortcut for the current directory
      - dotdot is a shortcut to the parent directory
  - -R: list subdirectories recursively
  - -S: sort by file size, largest first
  - -X : sort alphabetically by entry extension

cd

- Changes the current working directory
  - Absolute paths:
    - “/” at the start of your path means “starting from the root directory”
    - (”~”) at the start of your path means “starting from my home directory”
  - Relative Path: Starts from the current directory
    - e.g. ../folder (moves up one directory)
- Syntax: cd [directory]
  - Directory you want to navigate to
  - If omitted, cd defaults to the home directory
- Use Case: efficient file system navigation
  - Enables users to work effectively within different directories

- Key Options:
  - No Options: takes you to your home directory
  - .. : Moves you up one directory level
  - - : Switches to the previous directory
  - ~ : Represents home directory, useful for quickly navigating there
- “Tab” for auto filling
  - Applies to all commands, not just cd!

Demo

ls, cd

mkdir

- Creates new directories
  - Directories can be created using either absolute paths (starting from /) or relative path
  - Directory names can include special characters
    - Best to avoid spaces and stick to alphanumeric characters and underscore
- Use Case: Command helps create directories for organizing files in new projects

- Syntax: mkdir [OPTIONS] [DIRECTORY]
  - Takes one or more directory names as its arguments
  - If the directory already exists and if you don’t use -p, mkdir will return an error
- Key Options:
  - -p : creates the directory only if it doesn’t exist, makes parent directories as needed
  - -v: v stands for verbose, displays a message for each directory that is created

rmdir

- Removes empty directories from the file system
  - If the directory contains any files or subdirectories, it cannot be removed with rmdir
    - Will return an error
- Syntax: rmdir [options] directory_name
  - -p option: removes the specified directory and its parent directories if they are empty
- Use case: clean up empty directories left after moving or deleting files

- Comparison with rm -r:
  - rm -r command can remove directories that contain files or subdirectories
  - Use rmdir when you want to ensure that only empty directories are deleted

Demo

mkdir, rmdir

cp

- Copies files and directories from one location to another
- Syntax: cp [options] source destination
  - source: Files or directories you want to copy
  - destination: Location where you want to copy the file or directory
  - source can be one, or more files or directories, and destination can be a single file or directory
    - When multiple files or directories are given as a source, the destination must be a directory
  - If the destination file already exists, cp will overwrite it without warning

- Usage:
  - Create backups of important files or directories
  - Create a copy of a file before making changes with original as a reference
- Key Options
  - -i : Prompts before overwriting an existing file
  - -r : Copies an entire directory and its contents, including subdirectories
  - -v : (verbose mode) Displays the files being copied, useful for tracking the operation

mv

- Moves or renames files and directories
  - Very similar in spirit to cp, except moves instead of copying
  - mv copies the file to the new location and then deletes the original
- Syntax: mv [options] source destination
- Use Case: Move or rename files and directories to improve organization

- Key Options:
  - -i (Interactive): Prompts for confirmation before overwriting an existing file
  - -f (Force): Forces the move operation without prompting for confirmation, even if it involves overwriting files
  - -u (Update): Moves the source file only if it is newer than the destination file or if the destination file does not exist
  - -v (Verbose): Provides detailed information about the files being moved, including the source and destination paths
  - -n (No Clobber): Prevents overwriting of existing files. If a destination file exists, the move is not performed

rm

- Removes (deletes) files and directories from the file system
- Syntax: rm [options] file_or_directory
  - file_or_directory: file or directory you want to delete
- Use case: Managing disk space and keeping file systems organized
  - Clean up temporary files, remove old backups, or clear out directories

- Key Options
  - -f (Force): Forces the removal of files without prompting for confirmation
    - Useful for removing write-protected files
  - -i (Interactive): Prompts for confirmation before deleting each file
  - -r or -R (Recursive): Deletes directories and their contents recursively
    - This option is required for deleting non-empty directories
  - -v (Verbose): Displays detailed information about the files being deleted
  - -d (Directory): Removes empty directories.

Demo

cp, mv, rm

Outline

- ~~File and Directory Commands~~
- File Viewing and Editing Commands
- Commands for File Analysis
- Process Management
- Security and Permissions

File Viewing and Editing Commands

LINUX TERMINAL FOR BEGINNERS

https://www.reddit.com/r/linuxmemes/comments/oybfil/cat/?rdt=54918

File Viewing and Editing

- echo
- touch
- cat
- less and more
- head and tail
- Editors: vi, nano and gedit

echo

- Displays a line of text or string to the standard output (usually the terminal)
  - Similar to the print function in many programming languages
- Syntax: echo [options] [string]
  - string: the text or variables you want to display
- Use case: Widely used in scripting for providing user feedback, logging, and output formatting

- Key Options
  - -n (No Newline): No newline added at the end of the output
  - -e: Enables interpretation of escape characters (like \n, \t, etc.).
  - -E : Disables interpretation of backslash escapes (default behavior)
- What did echo say to the printf command?
  - "Relax, not everything has to be formatted perfectly!"

touch

- Creates an empty file or updates the timestamp of an existing file
- Syntax: touch [options] file_name
  - file_name: Name of the file to be created or whose timestamp is to be updated
- Common Use case:
  - Creating Empty Files
    - Set up placeholder files for configuration or testing
  - Updating File Timestamps
    - Update the last accessed or modified time of a file
      - Does not alter its content

- Key Options
  - -a : Updates only access time without changing the modification time
  - -m : Updates only the modification time without changing the access time
  - -t : Allows you to specify a particular timestamp instead of using the current time
    - touch -t [[CC]YY]MMDDhhmm[.ss] file.txt
  - -c or --no-create: Prevents touch from creating a file if it does not already exist
    - Only updates timestamps if the file exists

- How to know access and modification times?
  - ls -l --time=atime filename (access time)
  - ls -l filename (modification time)
  - Another alternative: stat
    - stat filename
    - Provides detailed information about a file
- Why did touch go to therapy? :-)
  - To work on its commitment issues — it kept making files but never opened up to them

Demo

echo, touch

cat

- Display the contents of files, combine multiple files into one, and create or append to files
  - Why named cat? can combine (concatenate) outputs also
- Syntax: cat [options] [file1] [file2] …
- Use case: quickly view file content or combine several files into a single output

- Key Options
  - -n : Numbers all lines in the output
  - -b : Numbers only non-empty lines
  - -v : Displays non-printable characters in a visible format
  - -s : Compresses multiple consecutive blank lines into a single blank line.

less

- A file viewer that allows you to view the contents of a file one screen at a time
  - Unlike cat, does not load the entire file at once
  - More efficient for viewing large files
- Syntax: less [options] filename

- Use Case: Efficiently view large files
  - Move up and down through a file using keyboard
    - Space for down, b for up; arrow buttons for scroll
  - Search for specific text within the file using / followed by the search term
    - To go to the next occurrence of the search term, press n
    - To go to the previous occurrence, press N
  - q: quit less and return to the command prompt
- Key Options
  - -N : Displays line numbers alongside the file content
  - -i : Makes search case-insensitive

more

- Used for viewing files one screen at a time, similar to less, but with fewer features
  - Primarily used for paginated output of file content
  - Use space to go forward by one screen, b to move one full screen up.
  - Press q to exit more and return to the command prompt

- Syntax: more [options] filename
- Key Options
  - -n : Defines the number of lines to display at a time
  - +number : Starts viewing the file from a specific line number.
- Less supports search, more doesn’t

Demo

cat, less, more

head

- Used to display the first few lines of a file or a group of files
  - By default, shows first 10 lines
  - But can specify number of lines or bytes to display
- Use case: Quickly view the beginning of a file without opening the entire content

- Syntax: head [options] filename
- Key Options
  - -n: Displays a specific number of lines from the start of the file
  - -c : Displays a specific number of bytes from the start of the file

tail

- Used to display the last part of a file
  - By default, shows the last 10 lines
  - But can customize to display a specific number of lines or bytes
- Use case: Monitoring log files in real-time or seeing the most recent content added to a file

- Syntax: tail [options] filename
- Key Options
  - -n : Displays a specific number of lines from the end of the file
  - -c : Displays a specific number of bytes from the end of the file
  - -f : Continuously outputs new lines as they are added to the file

- less: Best for interactive, full-file viewing with navigation and search capabilities
  - Ideal for exploring and reading large files
- more: Best for viewing files one screen at a time with limited interaction
  - Allows forward scrolling, making it more suitable for sequential reading
- head/tail: Best for quick, non-interactive previews of the start/end of a file
  - Simple, with no navigation or scrolling

Demo

head, tail

vi

- Opens vi editor, a powerful text editor available on most Unix-like systems
  - A modal editor that operates in different modes
  - Some learning curve, onced overcome, allows for very fast text editing
  - Particularly useful in environments where a GUI is not available!
  - An enhanced version of vi is vim (stands for "Vi IMproved")

- Syntax : vi filename
- Key Modes
  - Command Mode: default mode to navigate, delete, copy, and execute commands
    - Press i to enter insert mode
  - Insert Mode: Used for inserting or editing text
    - Any keystrokes in this mode are added directly to the file

Key Commands

- Saving and Exiting
  - Save and Exit: :wq
  - Exit Without Saving: :q!
  - Save Without Exiting: :w
- Navigation
  - Move to Start of Line: 0
  - Move to End of Line: $
  - Move to Start of File: gg
  - Move to End of File: G

- Editing
  - Delete a Line: dd
  - Copy a Line: yy
  - Paste: p
  - Undo: u
  - Redo: Ctrl + r
- Searching
  - Search Forward: /search_term
  - Search Backward: ?search_term
  - Repeat Search: n (for next occurrence)

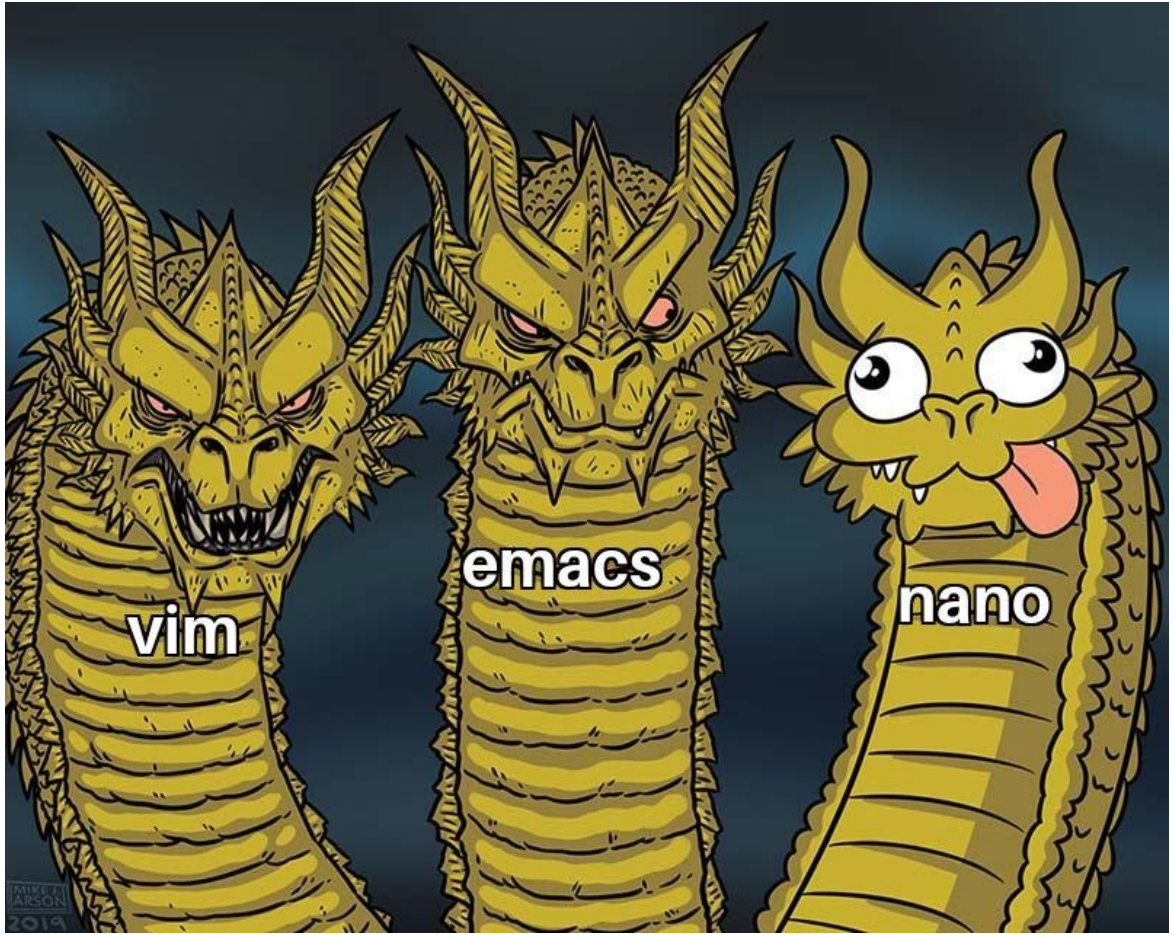
nano

- Opens a simple, easy-to-use text editor available on most Unix-like systems
  - Unlike vi or vim, nano is designed to be user-friendly
  - Keyboard shortcuts displayed at the bottom of the screen, making it more accessible for beginners
- Syntax : nano [options] filename
  - nano -c filename (enables line numbers)

gedit

- Default text editor for the GNOME desktop environment
  - Simple, user-friendly, and accessible GUI based editor
    - Vi and nano are terminal-based editors
  - Menus for saving, searching etc
  - Syntax highlighting for many programming languages
  - Can open multiple files in tab and switch between documents
  - Autosave and backup features to prevent data loss.
- Syntax : gedit [options] [filename]

Demo



https://pbs.twimg.com/media/Eb3V8MGX0AEWbRa.jpg

References

- The Linux Command Line by William Shotts
  - https://linuxcommand.org/tlcl.php
- https://ubuntu.com/tutorials/command-line-for-beginners#1-overview
- https://linuxize.com/ (good resource, use search box for info on different commands!)

Misc.

- File System:

https://www.linuxfoundation.org/blog/blog/classic-sysadmin-the-linux-filesystem-explained

- Figure of Unix variants: https://sosheskaz.github.io/technology/2017/05/12/Adventures-In-Bsd.html

Advanced Unix Commands

Kameswari Chebrolu

Outline

- ~~File and Directory Commands~~
- ~~File Viewing and Editing Commands~~
- Commands for File Analysis
- Process Management
- Security and Permissions

File Analysis Commands

Commands

- wc
- regex
- grep
- find
- cut

- paste
- sort
- uniq
- zip/tar
- redirection (>, >>, <)
- Pipe (|)

https://preview.redd.it/yjtwtofkxgy51.jpg?width=640&crop=smart&auto=webp&s=166b65dac9fb037c6d569744d12adbd3d84491ea

wc

- wc's motto: Every word counts!
- Counts the number of lines, words, and characters in a file or input from standard input
  - Will tell you if your file is too long, too short, or just right :-)
- Use Case:
  - Quickly obtaining statistics about text files
  - Often combined with other commands using pipes to process and analyze text
- Syntax : wc [OPTION] [FILES]
  - [FILES]: File(s) you want to analyze
    - If no file is provided, wc reads from standard input

- Output of wc typically consists of three numbers (when no specific option is used)
  - Number of Lines: Total number of lines in the file
  - Number of Words: Total number of words
  - Number of Bytes: Total size of the file in bytes
- Key Options
  - -l: Count lines
  - -w: Count words
  - -c: Count bytes
  - -m: Count characters
  - -L: Print the length of the longest line (in characters)

Demo

wc

https://xkcd.com/208/

Regular Expressions (regex)

- regex: a pattern that matches a set of strings
  - Used in text editors, programming languages, and command-line tools
- Metacharacters: characters with special meaning
  - "^" beginning of a line (Can also mean "not" if inside [])
  - "$" end of line
  - "." match any single character
  - "\" escape a special character
  - "|" or operation i.e. match a particular character set on either side

Quantifiers: specifying the number of occurrences of a character

- “*” Match the preceding item zero or more times
- “?” Match the preceding item zero or one time
- “+” Match the preceding item one or more times
- “{n}” Match the preceding item exactly n times
- “{n,} Match the preceding item at least n times
- “{,m}”Match the preceding item at most m times
- “{n,m}” Match the preceding item from n to m times

Groups and Ranges

- “ ( )” group patterns together
- “{ }” match a particular number of occurrences (seen before)
- “[ ]” match any character from a range of characters
  - ab[xyz]c "abxc" and "abyc"and "abzc"
  - [^…..] matches a character which is not defined in the square bracket
  - [a-z] matches letters of a small case from a to z
  - [A-Z] matches letters of an upper case from A to Z
  - [0-9] matches a digit from 0 to 9.

grep

- Grep: Global Regular Expression Print
- Searches for specific patterns within files or input provided via standard input
  - Used for text searching and processing
- Syntax : grep [OPTIONS] PATTERN [FILE...]
  - [OPTIONS]: Optional flags modify the behavior of grep
  - PATTERN: The regular expression pattern to search for
  - [FILE]: One or more files to search
    - If no file is specified, grep reads from standard input

- Key Options
  - -i: Ignore case (case-insensitive search)
  - -v: Invert match (show lines that do not match the pattern)
  - -r or -R: Recursively search directories
  - -n: Show line numbers with matching lines
  - -c: Count the number of matching line

- -H: Print the filename for each match
  - Useful when searching multiple files
- -o: Print only the matched parts of a line
- -E: Use extended regular expressions
- -w: match only whole words
- -A: Displays lines of text that appear after the matching line
- -B: Displays lines of text that appear before the matching line
- -C: Displays lines of text that appear both before and after the matching line

Demo

grep

find

- Used to search for files and directories based on various criteria
  - Can search for files by name, size, type
  - Can perform actions (execute commands) on found files
- Use case: Locate specific files, clean up old files, or performing actions on files that match certain conditions

- find [PATH] [OPTIONS] [CRITERIA] [ACTIONS]
  - [PATH]: The directory or directories to start the search from (default is the current directory)
  - [OPTIONS]: Optional flags that modify the behavior of find
  - [CRITERIA]: Conditions used to match files (e.g., by name, size, type)
  - [ACTIONS]: Actions to perform on the matched files (e.g., print, delete)

- Key Options and Criteria
  - -name: Search for files by name
  - -iname: Case-insensitive search for files by name
  - -type: Search for files by type
    - f: Regular file
    - d: Directory
  - -size: Search for files by size
    - +: Larger than
    - -: Smaller than
    - c: Size in bytes.

- -perm: Search for files or directories based on their permissions
- -mtime: Search for files based on modification time
  - +: More than n days ago
  - -: Less than n days ago
  - n: Exactly n days ago
- -exec: Execute a command on each found file
  - -delete: Delete files that match the search criteria
  - -print: Print the path of each found file (default action)

Demo

find

cut

- Used to extract specific sections of text from each line of input data
  - Useful for processing and filtering columns of data from text files, logs, or command output
  - Effective with structured data, such as CSV files or delimited text,
- Syntax: cut [OPTIONS] [FILE...]
  - FILE...: The file(s) to process
    - If no file is specified, cut reads from standard input

- Key Options
  - -f: Specifies the fields to be extracted
    - Fields are separated by a delimiter (tab is default)
  - -d: Defines the delimiter that separates fields in the input data
    - Default behavior: use the input delimiter as the output delimiter
  - -c: Extracts specific characters from each line of the input
  - -b: Extracts specific bytes from each line of input
  - --complement: Complement the selection
    - Displays all bytes, characters, or fields except the selected
  - --output-delimiter: Allows to specify a different output delimiter string

Demo

cut

paste

- Used to merge lines of files horizontally, creating columns of data
  - Combines corresponding lines from each file specified as arguments, separating them by a delimiter (which defaults to a tab)
- Use case:
  - Useful for joining data from multiple files or streams
    - Creates side-by-side comparisons or concatenated outputs
  - cat command merges files vertically (one after the other)
  - paste merges files horizontally, placing lines from different files side by side

- Syntax: paste [OPTIONS] [FILE...]
  - FILE...: The files to be merged
    - If no files are specified, paste reads from standard input
- Key Options
  - -d: Specifies a custom delimiter to use between merged lines
  - -s: Merges lines from one file sequentially, rather than in parallel with other files.
  - -: Indicates that standard input should be used in place of a file.

Demo

paste

sort

- Used to arrange lines of text files or input data in a specific order
  - By default, sorts lines alphabetically or numerically based on the first character, but can be customized
- Use case: Organize data for better readability, prepare data for further processing
- Syntax : sort [OPTIONS] [FILE...]
  - FILE...: The files to be sorted
    - If no files are specified, sort reads from standard input

- Key Options
  - -n: Sorts numerically, treating the first part of each line as a number
    - Useful for sorting lists of numbers or data that includes numeric fields.
  - -r: Reverses the sort order
  - -k: Sorts based on a specific field within each line
  - -t: Specifies a custom delimiter that separates fields in the input.
  - -u: Removes duplicate lines from the output, showing only unique entries
  - -M: Sorts lines based on the first three characters of the month name

Demo

sort

uniq

- Used to filter out or report repeated lines in a file or input data
  - Only works on adjacent lines
    - Identifies or removes duplicates that are directly next to each other
  - Does not perform any fuzzy matching
  - Only identifies lines that are exactly the same
- Use case: Commonly used in combination with sort to process sorted data

- Syntax : uniq [OPTIONS] [INPUT] [OUTPUT]
  - INPUT: The file to be processed
    - If no input file is specified, uniq reads from standard input
  - OUTPUT: The file where the results will be written
    - If no output file is specified, results are written to standard output

- Key Options
  - -c: Prefixes each line with the number of times it appears in the input
    - Useful for counting occurrences of each line
  - -d: Displays only the lines that are repeated (duplicates)
  - -u: Displays only the lines that are unique, excluding all repeated lines
  - -i: Ignores case when comparing lines
  - -f: Skips a specified number of fields before performing comparisons
  - -s: Skips a specified number of characters before performing comparisons.

zip

- Used to create compressed archive files
  - Bundles multiple files and directories into a single .zip file
  - Supports both compression and file management tasks
    - File management: can add, update, and delete files within an archive
  - .zip files are supported on many operating systems
- Use case: Helps create backups, package and distribute files
- Syntax: zip [OPTIONS] ARCHIVE FILES…
  - ARCHIVE: The name of the output .zip file to create or update
  - FILES...: The files and directories to include in the archive.

- Key Options
  - -r: Recursively include directories and their contents
  - -u: Update an existing archive with new or changed files
  - -d: Delete files from an existing archive
  - -x: Exclude files or directories from the archive
  - -e: Encrypt the archive with a password
  - -s : splits a large archive into multiple smaller files

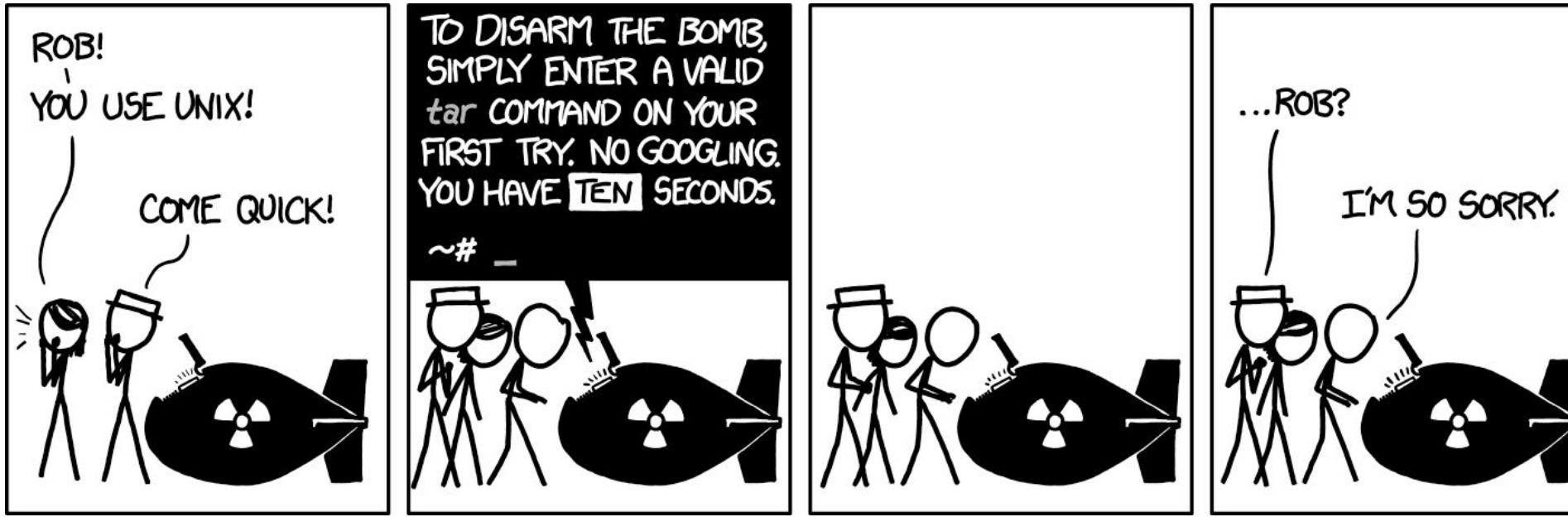
- Key Options
  - -d: Specify the directory to extract files into
  - -l: List the contents of the archive without extracting
  - -o: Overwrite existing files without prompting
  - -n: Never overwrite existing files
  - -x: Exclude specific files from extraction.

tar

- Used to create and manipulate archive files
  - Can bundle multiple files and directories into a single archive file, often with a .tar extension
  - Can extract files from archives
  - Supports various compression methods to reduce the archive using gzip, bzip2, or xz
  - Preserves file metadata such as permissions, ownership, and timestamps
    - Very useful for backups and transfers

- Syntax : tar [OPTIONS] [ARCHIVE] [FILES...]
  - ARCHIVE: The name of the archive file to create, extract, or manipulate
  - FILES...: The files and directories to include in the archive or extract from it.

- Key Options
  - -c: Create a new archive
  - -x: Extract files from an archive
  - -t: List the contents of an archive without extracting
  - -f: Specifies the archive file name
    - Must be followed by the name of the archive file
  - -z: Compress or decompress using gzip
  - -j: Compress or decompress using bzip2
  - -J: Compress or decompress using xz
  - -v: Verbose mode, displays the progress of the operation



https://xkcd.com/1168/

- Zip:
  - Combines both compression and archiving in a single step
  - Often used for cross-platform compatibility
- Tar:
  - Primarily used for bundling files, with compression applied separately
  - Favored for its efficiency and detailed preservation of file attributes

Input/Output

- A process is an instance of a program that is being executed
- Stream: a special file that either continuously receives text in or pushes text out
- When you run a command, OS creates a process to execute that command

- Whenever a process starts, process is given access to three “standard” streams
  - fd is file descriptor
  - Standard input (stdin; fd is 0)
  - Standard output (stdout, fd is 1)
  - Standard error (stderr, fd is 2); used when an error has occurred
- When a process starts
  - stdout and stderr are configured to print whatever they receive to the terminal screen
  - stdin is configured to read input from the user’s keyboard

Redirection

- Redirection controls where the output of a command goes and where input comes from
  - Allows you to redirect standard input (stdin), standard output (stdout), and standard error (stderr)
- > (Output Redirection) : Redirects the standard output (stdout) of a command to a file
  - If the file already exists, it will be overwritten
  - Syntax: command > file

- >> (Append Redirection) : Redirects the standard output (stdout) of a command to a file, but instead of overwriting, appends output end of the file
  - Syntax: command >> file
- < (Input Redirection): Redirects the standard input (stdin) for a command from a file
  - Instead of typing input directly into the command, it reads the input from the specified file
  - Syntax: command < file

- command > file same as command 1> file (stdout redirected, stderr still screen)
- command 2> file (send stderr to file, stdout is screen)
- command 2> error.txt 1> out.txt (send both to different files)
- command > file 2>&1 (send both to same file)
- command 2> /dev/null (suppress error messages)
  - /dev/null is a special file that discards anything written to it

pipe

- How many files and folders in /etc ?
  - ls /etc > temp.txt; wc -l temp.txt; rm temp.txt
- A pipe (|) allows the output of one command to be used as input for another command
  - Shell connects the stdout of the first command directly to the stdin of the second command
    - Operates entirely in memory
    - Unidirectional: flows from left to right

  - Enables chaining of multiple commands together to perform complex operations
- Use case: Process data through a series of commands without needing to store intermediate results in temporary files
- Syntax: command1 | command2 | command3 ...

Command Substitution

- How to use the output of a command in the arguments of another
  - Note: pipes are great for sharing stdin and stdout between processes
- Command substitution: Output of a command replaces the command itself
  - $(command) or
  - `command`

Outline

- ~~File and Directory Commands~~
- ~~File Viewing and Editing Commands~~
- ~~Commands for File Analysis~~
- Process Management
- Security and Permissions

Process Management

Commands

- ps
- pkill

Process Management

- Methods and tools to control, monitor, and interact with processes running on the system
- A process is a running instance of a program
  - Each process in Linux has a unique Process ID (PID)
  - From a process, another process can be created
    - Achieved via fork system call

- Parent-child relationship exists between the two processes
  - PID (Process ID): A unique identifier assigned to each process.
  - PPID (Parent Process ID): The PID of the process that started (or "parented") the current process.
- init has process id 1
  - Parent of all processes
  - Executed by the kernel during the booting of a system

- Process States:
  - Running: The process is currently executing
  - Sleeping: The process is waiting for an event (e.g., I/O completion)
  - Stopped: The process has been stopped, usually by receiving a signal
  - Zombie: The process has completed execution but still has an entry in the process table

ps

- Displays information about the currently running processes on the system
  - Default output is a list of the processes associated with the command-line
  - Shows unique process id (pid), terminal used, amount of CPU time, and the program name
- Use case: Commonly used to monitor running processes, check for specific processes, or troubleshoot system performance issues
- Syntax: ps [options]

- Key Options
  - -e or -A: Lists all processes running on the system
  - -f: Displays full-format listing
  - -u username: Shows processes owned by the specified user
  - -p PID: Displays information about the specified process ID(s)
  - -C command_name: Filters processes by the command name
  - aux: Displays detailed information about all processes
    - “a”: display the processes of all users
    - “u”: user-oriented format → more details
    - “x”: list the processes without a controlling terminal
      - Started on boot time and running in the background

- PID: Process ID - The unique identifier for the process.
- TTY: Terminal type associated with the process.
- TIME: Total CPU time the process has consumed.
- COMMAND: The command that started the process.
- USER: The user who owns the process (shown with options like aux).
- %CPU: The percentage of CPU usage (shown with options like aux).
- %MEM: The percentage of memory usage (shown with options like aux).
- VSZ: Virtual memory size (shown with options like aux).
- RSS: Resident Set Size, the non-swapped physical memory that the task has used (shown with options like aux).
- STAT: Process state (e.g., R for running, S for sleeping) (shown with options like aux).
- START: The start time of the process

pkill

- Used to send signals to processes to request actions like termination, suspension, or restarting
  - Targets processes based on their names or other attributes, rather than process ID (PID)
    - Kill another command which is similar but needs PIDs
- Use Case: Control processes by sending them specific signals
  - Request actions like termination, suspension, or restarting
- Syntax: pkill [options] pattern

- Key Options
  - -s SIGNAL: Specifies the signal to send to the matching processes
    - If omitted, the default signal is SIGTERM
    - To know what SIGNALs available, use kill -l
  - -f: Matches against the full command line of the processes, not just the process name
  - -u USER: Targets processes owned by the specified user
  - -t TTY: Targets processes associated with the specified terminal
  - -P PID: Targets child processes of the specified parent PID

https://i.redd.it/2ndv691za33a1.png

Outline

- ~~File and Directory Commands~~
- ~~File Viewing and Editing Commands~~
- ~~Commands for File Analysis~~
- ~~Process Management~~
- Security and Permissions

Security and Permissions

Commands

- su
- sudo
- Access control

Superuser

- Superuser: user with super powers
  - A real user account (often root) that can do just about anything (modify/delete files, run any programs etc)
- For security reasons, “su” was introduced
  - Can mean ‘superuser’ or ‘switch user’
  - Helps change to another user without having to log out and in
    - Terminal session switched to the other user
    - Requires password of the other user
  - Administrators spend most time using normal account, when needed switch to superuser, do task and logout

- Syntax: su [options] [username]
  - username: User you want to switch to
    - If not specified, command switches to root
- Key Options:
  - -c [command]: Executes a single command as the specified user and returns to previous user after command is run
  - -s [shell]: Specifies which shell to use when switching users
  - -p or --preserve-environment: Preserves the current environment variables instead of loading the new user’s environment

- Further improvement, “sudo” was introduced
  - “switch user and do this command”
    - Does not fully switch to that user’s environment
  - Asks for the current user's password (not root's)
    - Permissions for using sudo are defined in the /etc/sudoers file
      - Administrators can specify which users are allowed to run which command
    - Runs the command with elevated privileges
    - Entered password is cached for a default period (usually 15 minutes)
      - No need to re-enter it for subsequent sudo commands

  - Prevents long-lived terminal sessions with dangerous powers
- Use Case: Grants users temporary access to perform administrative tasks
  - Tasks otherwise restricted to root user or a system administrator
- Be very careful when using sudo or su

- Key Options:
  - -u [user]: Runs the command as a specified user, instead of root
  - -l: Lists the commands that the current user is allowed to run with sudo
  - -k: Invalidates the current user’s cached credentials, forcing sudo to prompt for a password again.
  - --preserve-env or -E: Preserves the current environment variables when running a command

- Why Use sudo Instead of su?
  - More Secure: Unlike su, sudo doesn’t require sharing the root password, which limits security risks
  - Granular Control: Administrators can limit which commands users can run with sudo
  - Auditability: Actions performed with sudo can be logged, helping track which users executed which commands

Access Control

- UNIX is a multi-user system
- Every file and directory (in your account) can be protected from or made accessible to other users. How?
- Permissions for a file or directory may be any or all of
  - r - read; w - write; x - execute
  - a directory must have both r and x permissions if the files it contains are to be accessed

- Each permission (rwx) can be controlled at three levels:
  - u (user = yourself)
  - g (group, a set of users)
  - o (others, everyone else)
- File access permissions are displayed using “ ls -l”
- Use Case: Sensitive information is protected and only accessible to authorized personnel

- First field: - for File, d for Directory, l for Link
- Second,third,fourth fields: permissions for owner, group and others
- Fifth field: specifies the number of links or directories inside this directory
- Sixth field: user
- Seventh field: group
- Eighth field: size in bytes (use -lh option for better understanding)
- Ninth field: date of last modification
- Tenth field: name of the file/directory

chmod

- “chmod” command helps change access permissions for files you own
  - chmod [OPTIONS] MODE FILE(s)/directory(s)
  - Only root, the file owner or user with sudo privileges can change the permissions of a file
- Use Case: Set appropriate permissions on files and directories of web server to ensure both security and functionality

- Symbolic Mode:
  - u - File owner.
  - g - Users who are members of the group.
  - o - All other users.
  - a - All users, identical to ugo
  - - Removes the specified permissions.
  - + Adds specified permissions
  - = Changes the current permissions to the specified permissions
    - If no permissions are specified after the = symbol, all permissions from the specified user class are removed
  - E.g. chmod g=r filename; chmod a-x filename;

- Numeric mode:
  - r (read) = 4
  - w (write) = 2
  - x (execute) = 1
  - no permissions = 0
  - chmod 640 file
    - octal notation, user has r+w, group has read, others have none
  - chmod -R 700 dirname
    - Recursively set read, write, and execute permissions to the file owner
    - No permissions for all other users on a given directory

References

- https://ubuntu.com/tutorials/command-line-for-beginners#1-overview
- https://linuxize.com/ (good resource, use search box for info on different commands!)